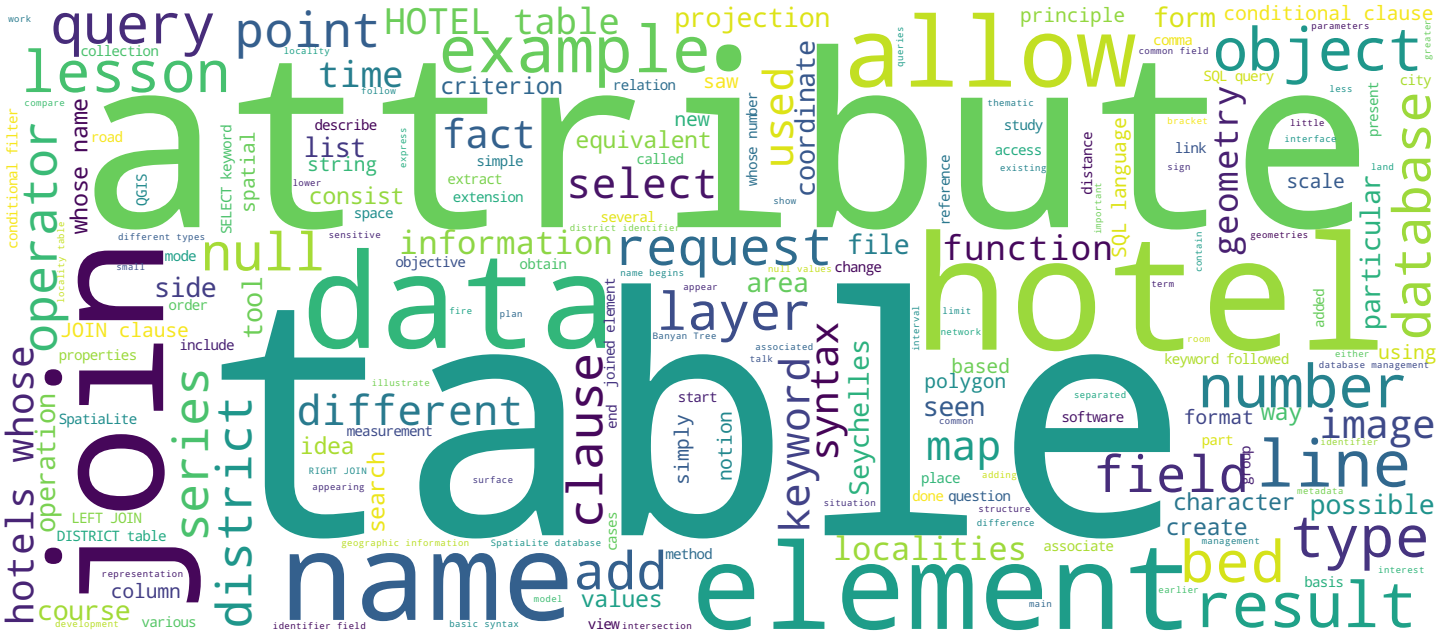
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

RESCIF
Réseau d'excellence des sciences de l'ingénieur de la Francophonie

Conditional queries

An Introduction to Geographic Information Systems

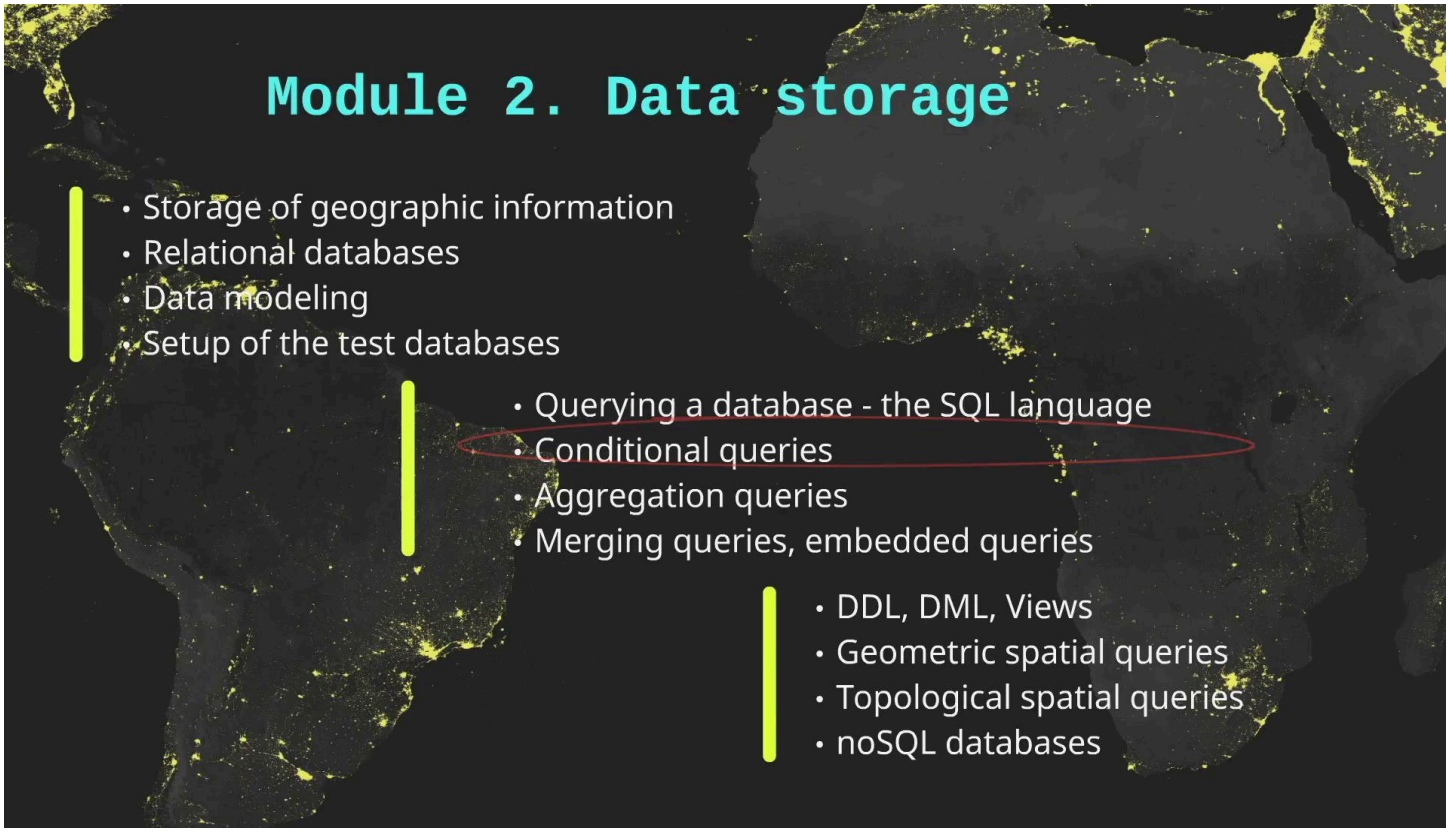
Stéphane Joost, Marc Soutter, Fernand Kouamé, Amadou Sall



Search MOOC

Video

EPFL



Welcome to this lesson on conditional queries.

Notes

Summary

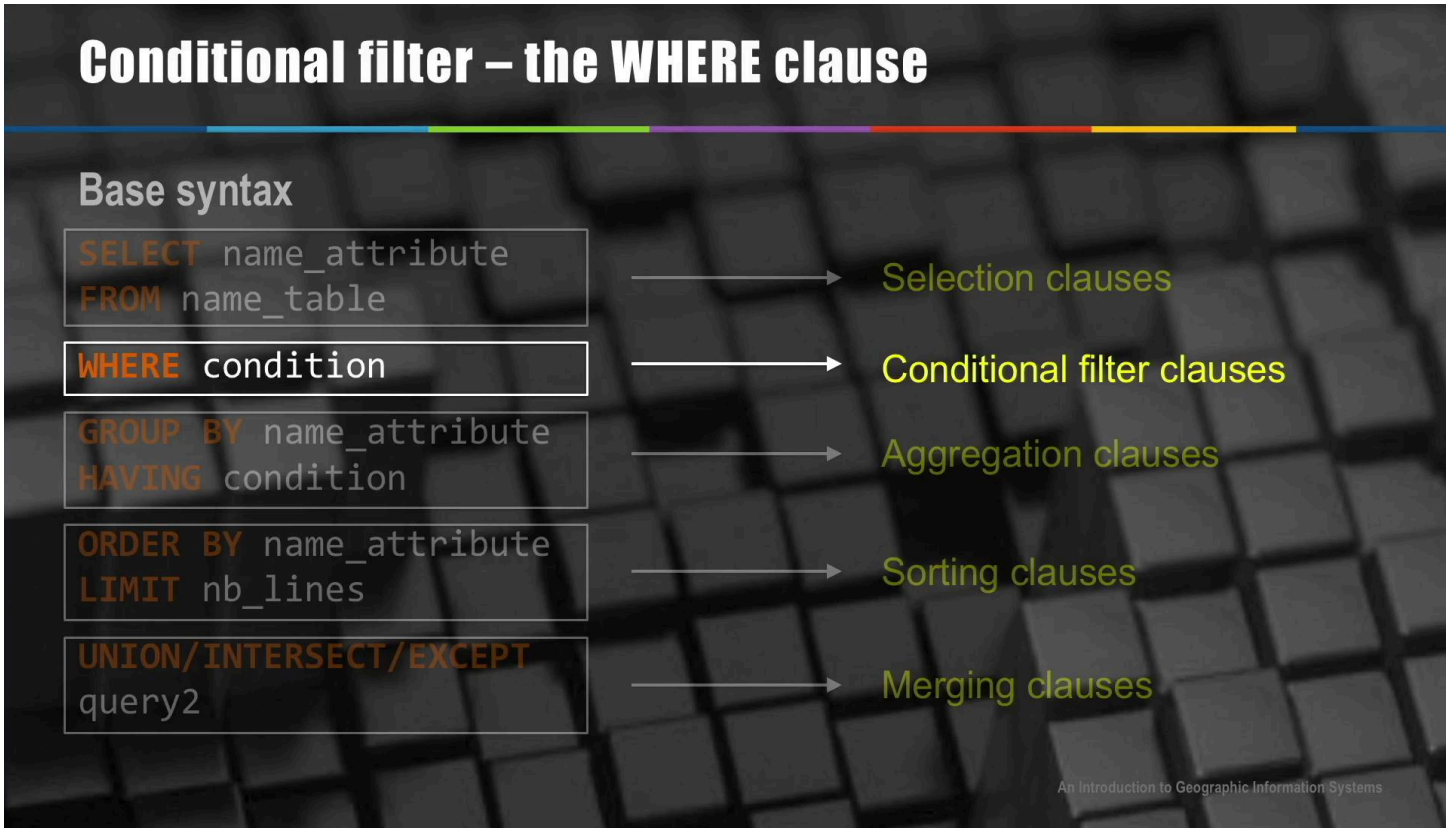
0m 04s

This type of query allows to extract a subset of a group of objects. For example, we can extract a set of red cars from a parking in which an enormous mass of vehicles are parked, but you can't imagine anyway! Even with mobility cars. Even with mobility cars. Mobility! And you have the card, and you have the mobility card and you can't find your car. And you when you say I need a red car... Sometimes in the cars, those who were not with us from the start, but which took the train to join our car later. Yeah, but formulate it in SQL because they can't understand anything here, they can't understand anything. We will discuss in this lesson the question of conditional requests, the requests that allow to extract data on the basis of an attribute criterion in the example that accompanies us throughout this course on the Seychelles, some requests that would allow for example to extract the subset of asphalted roads. The objective of the lesson is to study the principle of the syntax of a conditional query and to understand how these queries lead to the notion of join which allow to associate several tables, so that at the end of the lesson you are able to use, to write conditional queries, to select, to filter data on the basis of an attribute criterion and to use table joins.

Notes

Summary

0m 25s

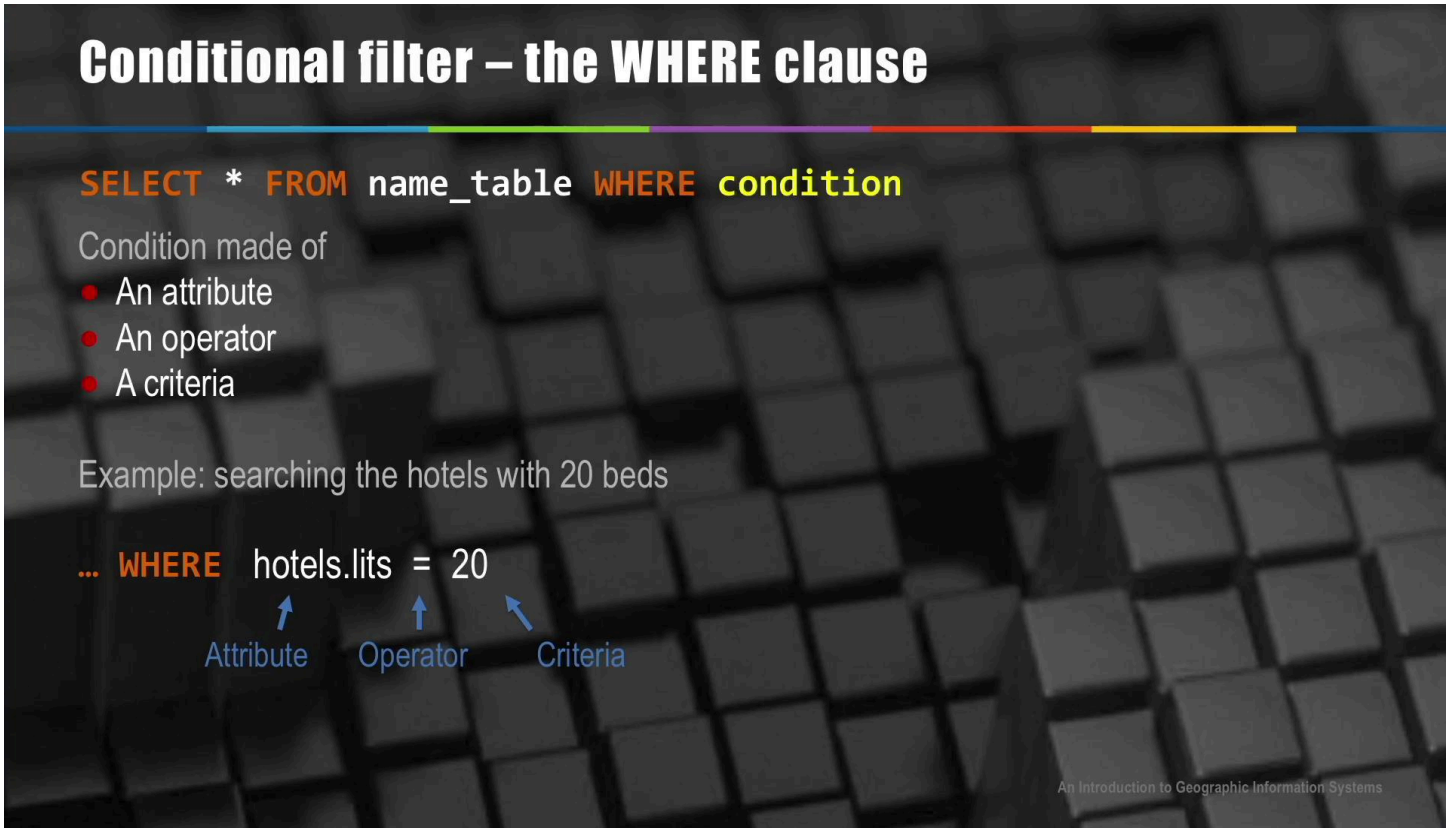


We will see in this lesson first the notion of conditional filter, based on the use of the WHERE clause in an SQL query then the various operators that can be used in these clauses, then the joins that rely on the WHERE clause and to conclude, joins based on another type of clause which is the JOIN clause. We saw in the previous lesson of introduction to the SQL language that it is based on a basic syntax which includes a certain number of keywords defining clauses, selection clauses, conditional filter, aggregation, etc. And in this lesson, we address the conditional filter clause which is expressed by the WHERE keyword with a condition.

Notes

Summary

1m 57s

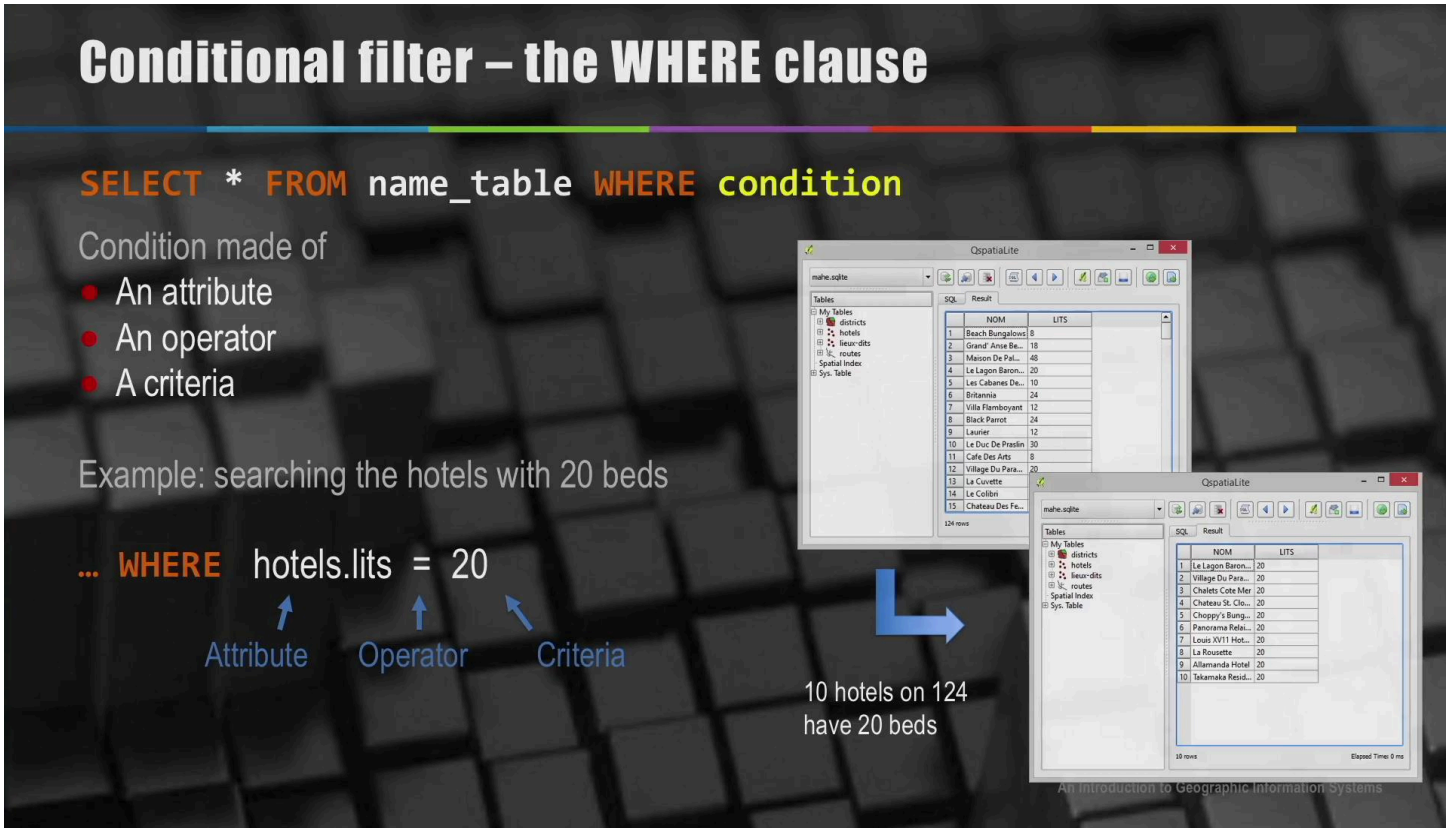


The basic syntax of this query includes first of all the word SELECT followed by the name of the attributes or the asterisk metacharacter when we want to select all the attributes, of the FROM keyword, followed by the name of the table from which the information will come and finally the WHERE keyword followed by the condition, a condition which includes 3 elements: an attribute, an operator and a criterion. If we take the example of the search for hotels with 20 beds in the Seychelles, wee see that the condition in the conditional clause includes as an attribute the "hotels.beds" keyword, so the attribute "beds", as the operator the "equal" sign and as a criterion the value 20.

Notes

Summary

2m 49s

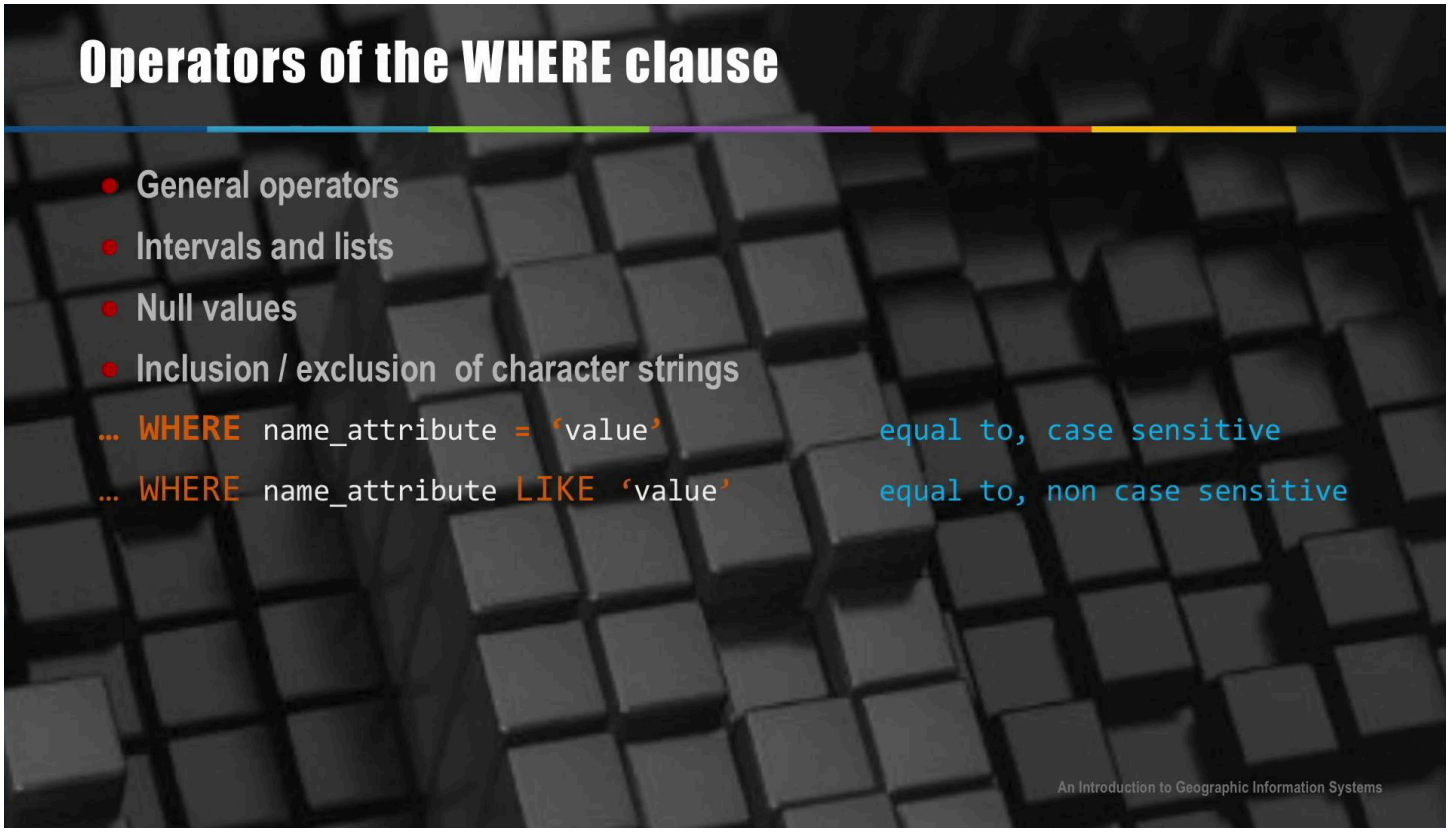


In the case of the Seychelles database, we have a set of 124 hotels and if this conditional filter request is applied on the criterion "number of beds = 20", we see that we are actually extracting a series of 10 hotels with exactly 20 beds.

Notes

Summary

3m 34s

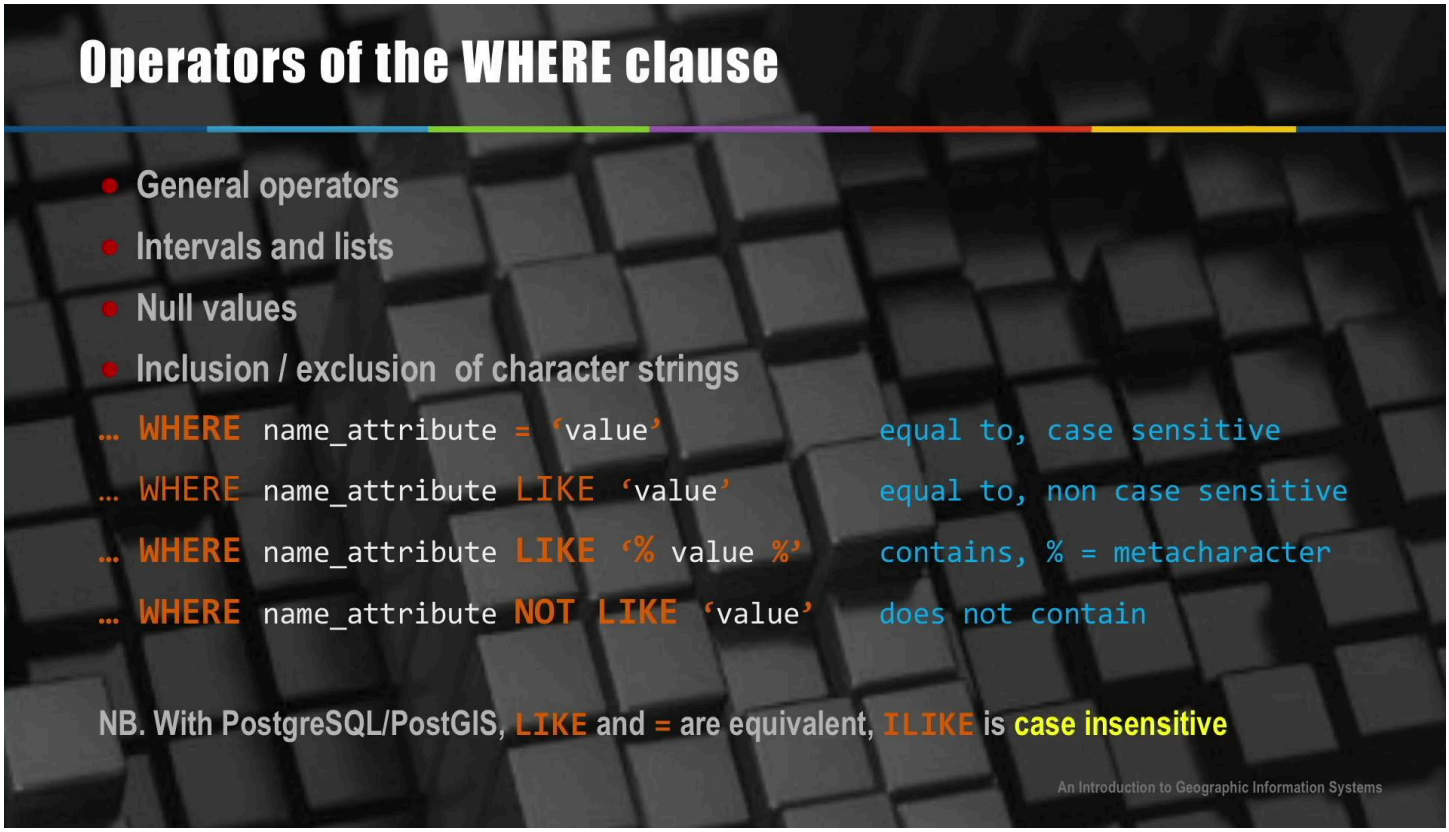


The operators that we can use in a WHERE clause are of different natures starting with the general operators and in the first place the "equal" operator which allows to compare two values between them and its alter ego the "different from" attribute that can be expressed either in the form of an exclamation point and the equal sign or 2 signs "smaller and bigger than". Then the 2 attributes "lower and higher than" or alternatively "less than or equal to". The intervals and lists, with for the lists the IN keyword, followed by a series of values separated by commas and enclosed in brackets which express the idea that the value of the attribute is present in the collection or the sample of the proposed values. And for the intervals, the keyword BETWEEN followed by 2 values separated by the AND keyword which expresses the idea that the value of the attribute is between the 2 limits defined by value 1 and value 2. The test on null values with the keywords: IS NULL / IS NOT NULL to check if the value of the attribute is null or not null. And finally, the inclusion and exclusion operators of strings of characters which allows to work on strings of characters with first the equality operator, which allows to compare a character string with a value in brackets.

Notes

Summary

4m 03s

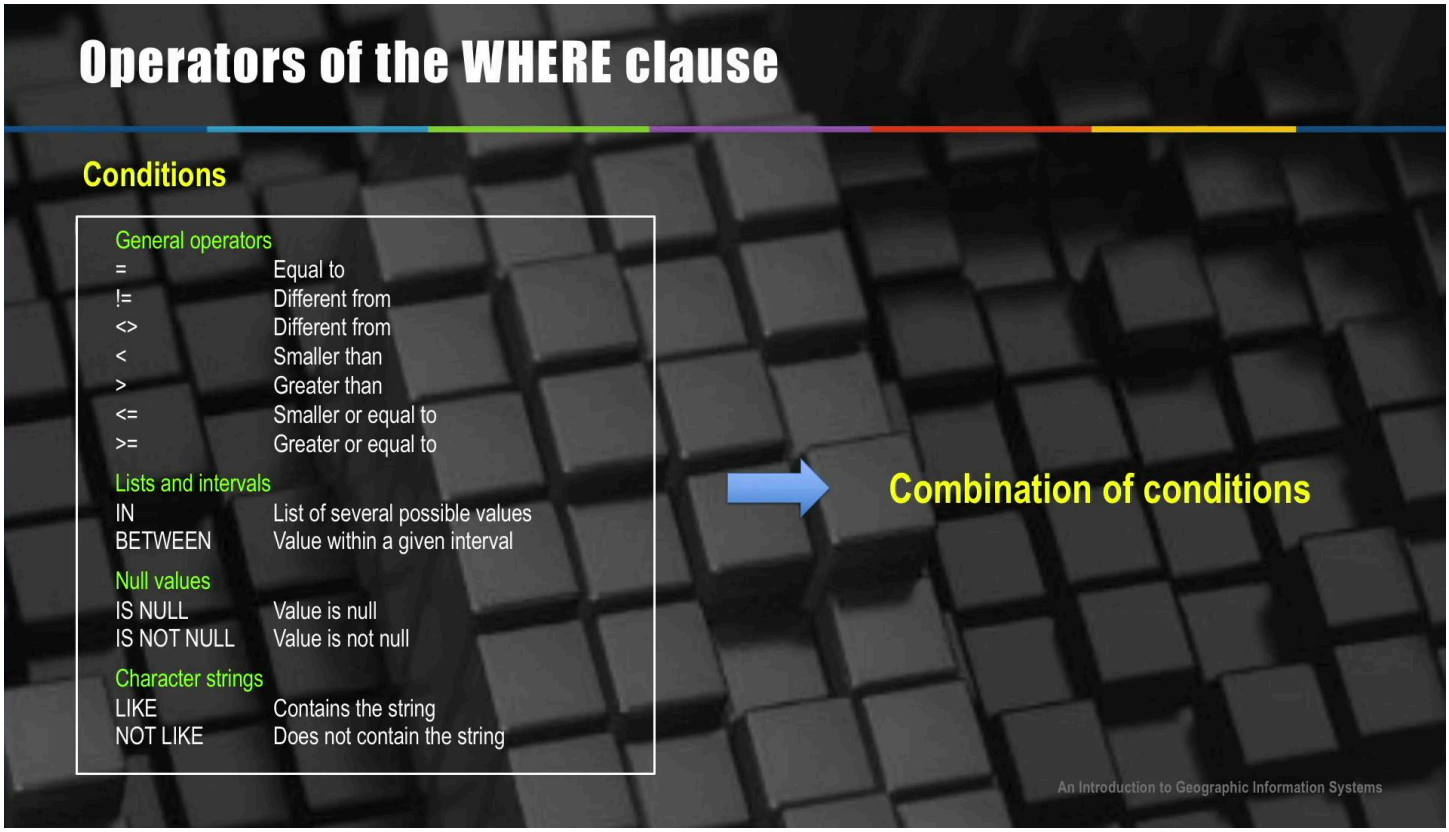


An equivalent of the equal sign is the LIKE operator except that it is case-insensitive, so independent from upper and lower case letters used to describe the value string of character this is true in the case of SQLite but it is not systematic, with all the database systems. The LIKE operator with the metacharacters, here in the SQL language, is the "percent" sign that is used as a metacharacter, so as a replacement character for strings of characters. So here, we will look for a set of objects whose attribute includes the value string of character in one place or another. And finally, the NOT LIKE operator to say that a value is not included. And as we say in PostgreSQL, LIKE and "equal" are equivalent and we have another keyword, another operator, ILIKE, which is case-insensitive.

Notes

Summary

5m 34s



This table summarizes the different types of operators we have just seen, so the general operators, lists and intervals, the null values and the operators for the strings of characters. These operators are used to define conditions and what is interesting in the WHERE clauses is that these conditions can be combined to do elaborated researches.

Notes

Summary

6m 39s

The combination of conditions is based on the AND and OR keywords which allow to associate 2 conditions inclusively or exclusively. Additional conditions can be added to this system by using the usual bracket rules.

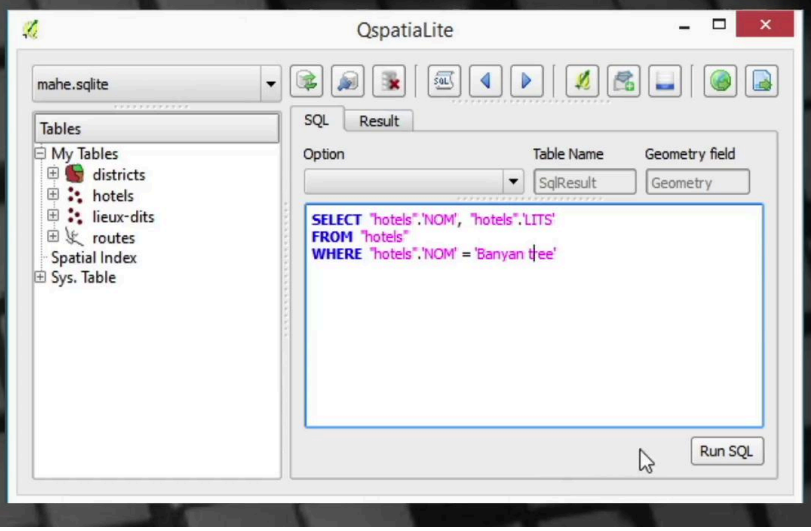
Notes

Summary

7m 03s

Operators of the WHERE clause

Combination of conditions



So we illustrate here these operators in the case of a SpatiaLite database by selecting in the HOTELS table the names and the beds and by adding as conditional clause the fact that the beds must be equal to 20 and we make this request. By modifying this query we can search for all the hotels with fewer than 20 beds, all the hotels that have more than 20 beds of course, all the hotels that have 158 or 176 beds and there are 2 which have exactly 158 and 176 beds. Alternatively, we can search for all the hotels whose number of beds is between these 2 values of 158 and 176 and we see that there are 3, which also allows to show that the limits, 158 and 176, are inclusive and not exclusive. We then search here for all the hotels whose name corresponds to Banyan Tree. We find this hotel.

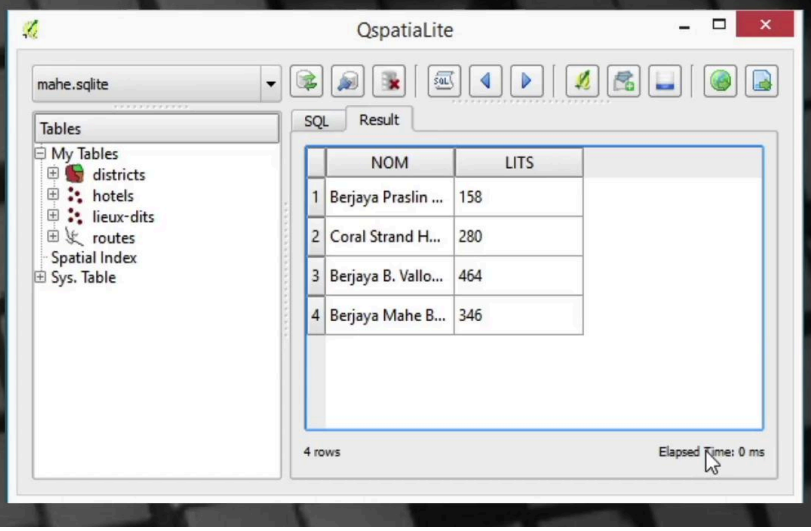
Notes

Summary

7m 26s

Operators of the WHERE clause

Combination of conditions



and we see that if we had written Banyan Tree with a lower case T, the request would have given no result whereas in the case of SpatiaLite with a LIKE, the query is not case-sensitive and we find the hotel wanted. We then research all the hotels whose name begins with B, and we find 13. Then all the hotels whose name begins with B or whose name begins with C. And we add an additional condition, the fact that the number of beds must be greater than 100. And we see that there are 4 candidates meeting these criteria.

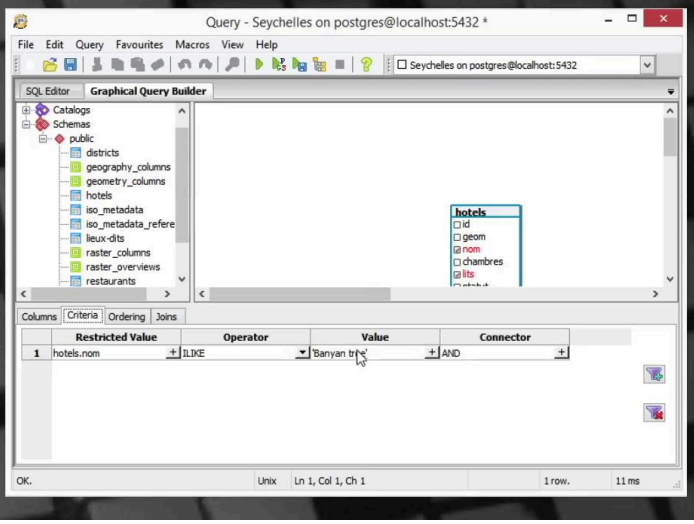
Notes

Summary

8m 53s

Operators of the WHERE clause

Combination of conditions



We now perform the same series of operations in the case of a PostGIS database by using the pgAdmin interface, so again we select in the hotel table the NAME and BEDS fields, we add a criterion in the graphic interface so that the number of hotels... the number of beds is equal to 20 and we change the operator in this request to have the number of hotels whose number of beds is less than 20, now greater than 20. As before, we will search for hotels whose number of beds is 158 or 176, and we see that we have to write in this interface the whole of the condition in the same way as in the SpatiaLite case where we do this in pure SQL. So here whether we use in this case the graphic interface or not does not make any difference. The request for hotels whose number of rooms is between 158 and 176, you saw it appear and now we look for the hotel which is called Banyan Tree that we find of course, the example with a lower case T and we see that indeed there is no result, by replacing the "equal" operator by the LIKE operator, still no result so as I told you LIKE is case-sensitive. We must use the ILIKE operator to make a request that is not case-sensitive.

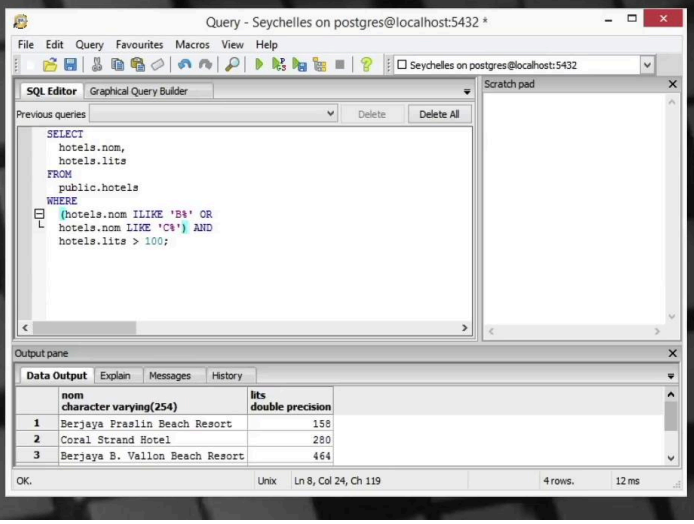
Notes

Summary

9m 39s

Operators of the WHERE clause

Combination of conditions

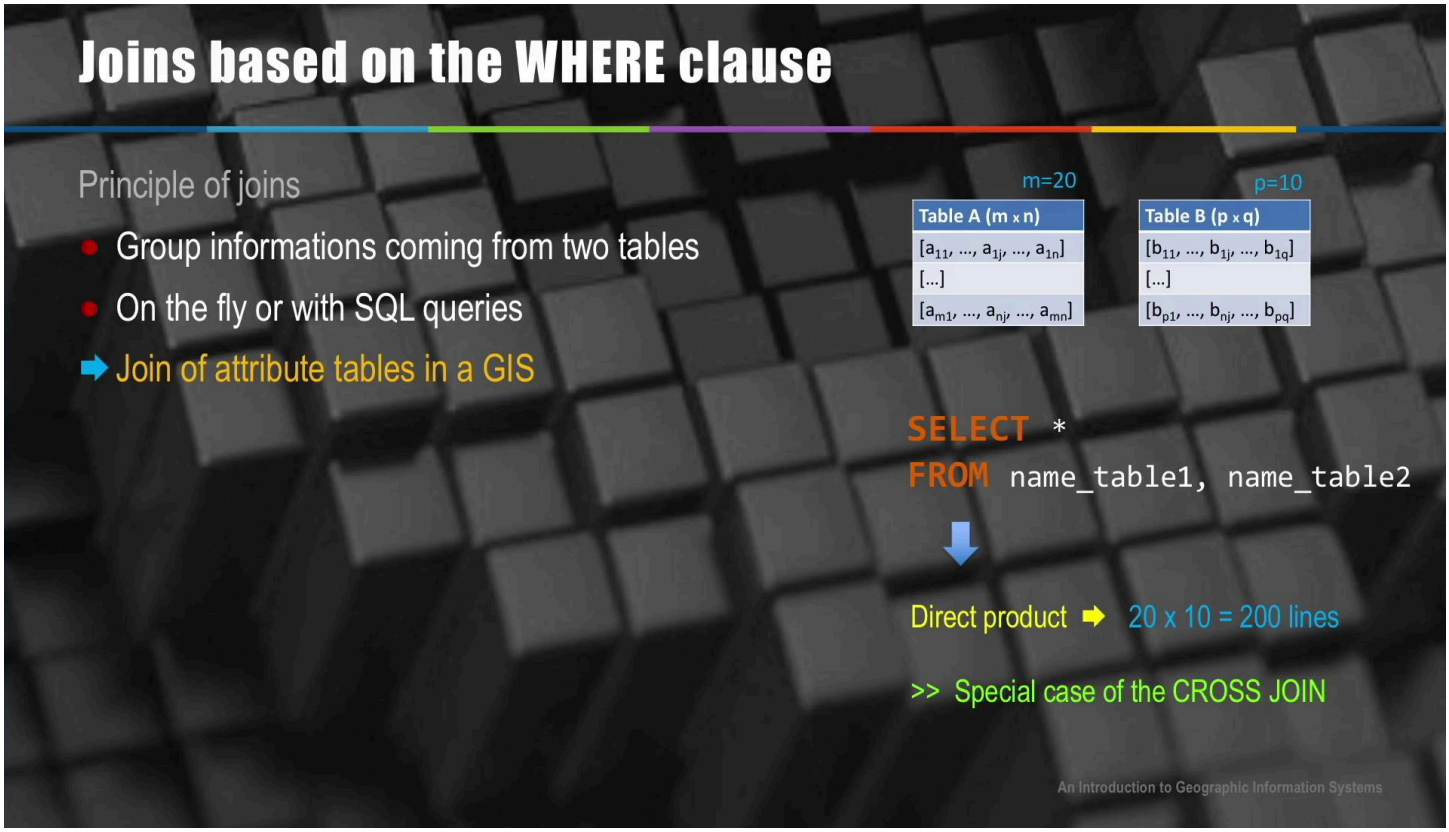


As before, we look for hotels whose name starts with B, we also find 13 which is reassuring, and we now associate an additional request with again the name of the hotel, which starts this time with the letter C and then third supplementary request, the number of beds greater than 100 as before. And as earlier, we find 4 candidates. No. Something curious happened here! The brackets in the right place, we find our 4 candidates.

Notes

Summary

11m 34s

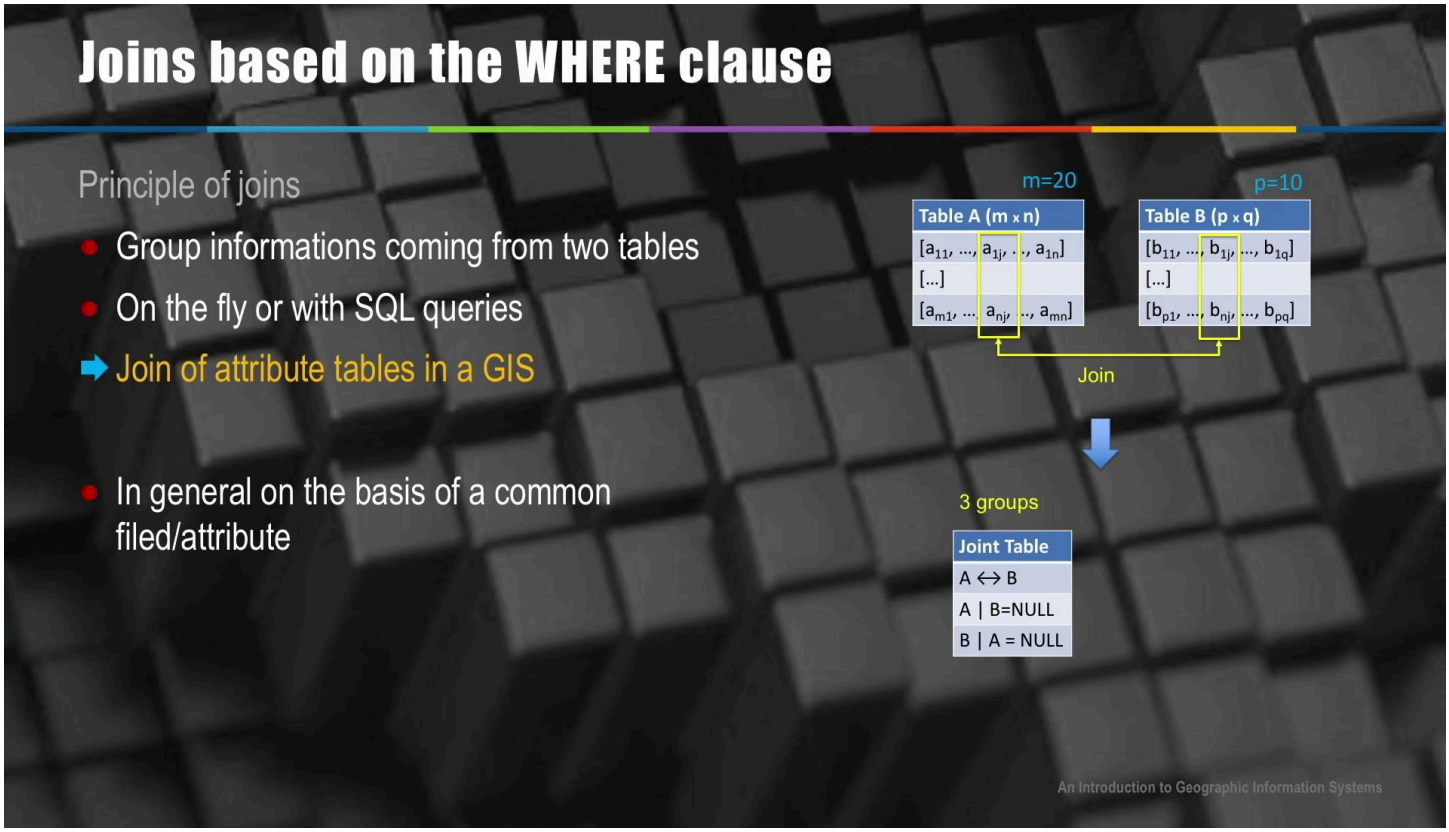


The principle of joins consists in associating information coming from 2 or more tables and can be applied either on the fly or using SQL queries. We saw in the lesson on data modeling, an example of an attribute table join on the fly performed in the QGIS software. Here we will look at the way to use the SQL language to perform these table joins. Let's suppose that we have 2 tables, A and B, consisting of a series of lines which contain objects, which themselves have a number of attributes from A1 to An and similarly for the table B which has dimensions P by Q and let's suppose that table A has 20 lines, and table B 10 lines. The association request for these standard tables would be of the SELECT type, the series of the name of the attributes, the FROM keyword and then the 2 table names that follow each other separated by a comma. This type of query would give the cartesian product of these 2 tables so each element, each line of table A would be multiplied by each line of table B or associated with each line of the table B so as to produce a set resulting in 200 lines. And this set resulting in 200 lines corresponds to the particular case, a particular case of join called the CROSS JOIN.

Notes

Summary

12m 31s



The table join is most often based on a common field which allows to associate together the rows of 2 tables for which this common field has the same value. The type of join leads to distinguish 3 types of situations in the results that we produce, in the comparisons between the lines of the 2 tables. First of all, the case where the 2 lines are joined by a common field which has the same value.

Notes

Summary

14m 12s

After we have the situations where the lines of table A which do not have join line equivalents in table B therefore B is null, and conversely the lines of table B which have no equivalent on the side of A therefore for which A is null. If we look at the size of the result tables obtained we see that if the number of joins is null, so there is no line of the 2 tables which have a common value for the JOIN field, we find a set of 30 lines in the result, so the 20 lines of table A for which B is null and the 10 lines of table B for which A is null. In the case of 5 matches, 5 lines of each of the tables joined together, we see that the lines of A which would have a null match on the side of B are 15 and conversely the lines of B which are null on the side of A are 5 and the final result would be a total of 25 lines in a query that would take up all of these values. And finally in the same spirit if we have 10 joins, we find 20 possible results in total. The general syntax of which a join based on the WHERE clause looks like this, so first the SELECT keyword followed by the attribute of the first table and the attribute of the second table separated by a comma.

Notes

Summary

14m 44s

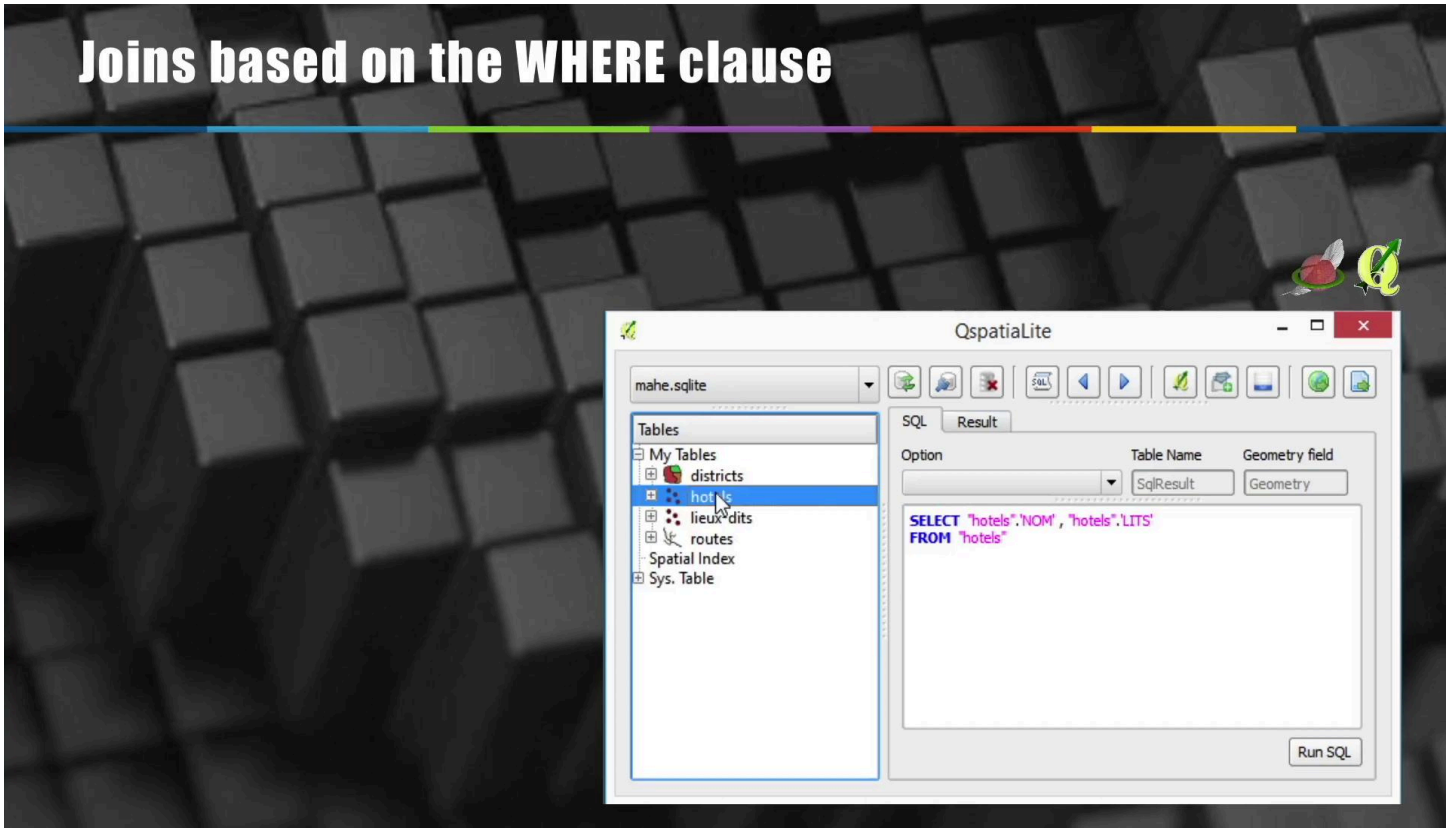
The FROM keyword, followed by the 2 table names separated by a comma again and then the WHERE clause with the condition that associates an attribute of the first table, the equal operator and as a criterion an attribute of the second table, the 2 attributes of the WHERE clause defining the join.

Notes

Summary

16m 23s

Joins based on the WHERE clause



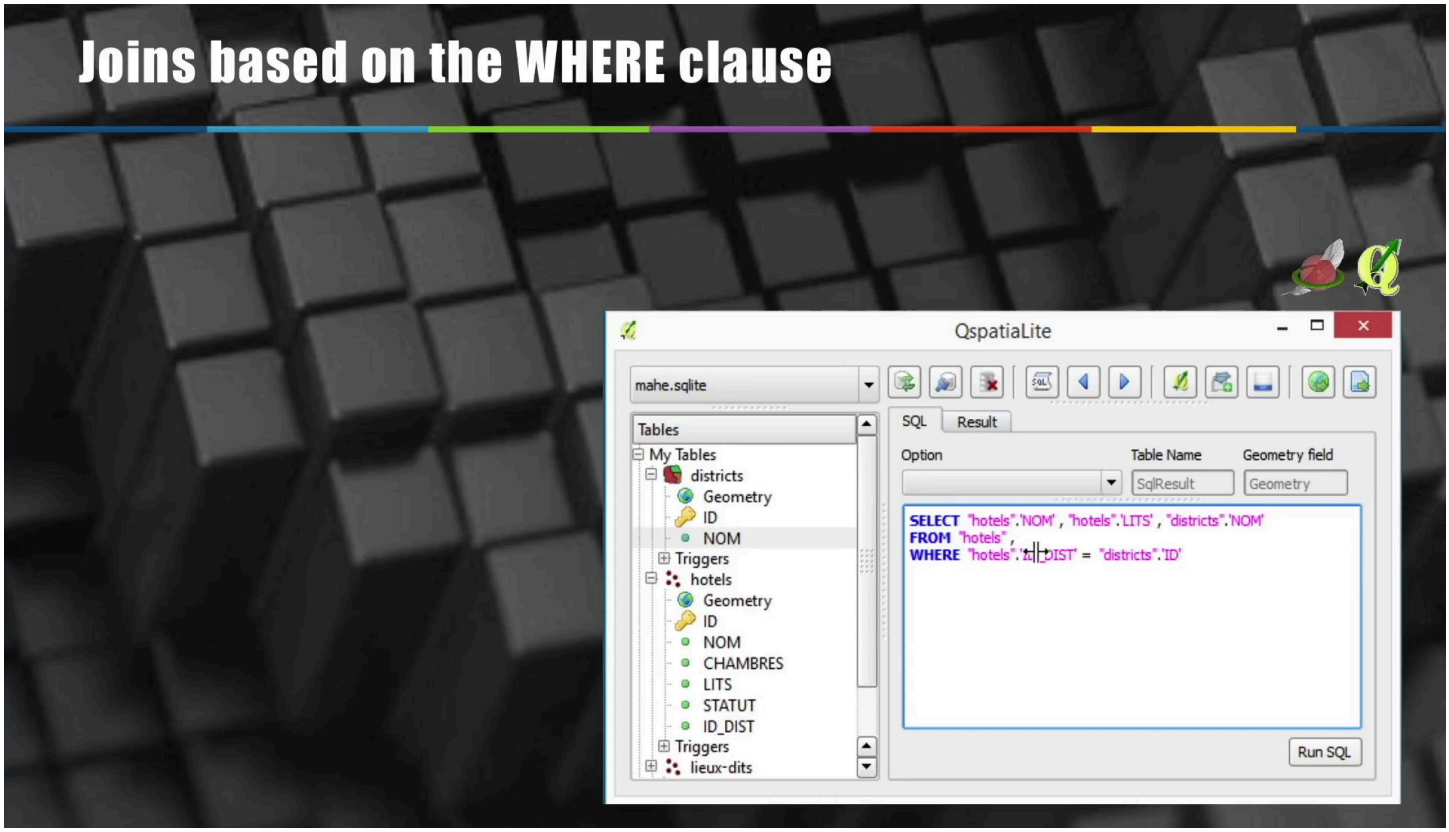
Taking the case of the SpatiaLite database on the Seychelles, we write the same query as before for hotel beds.

Notes

Summary

16m 52s

Joins based on the WHERE clause



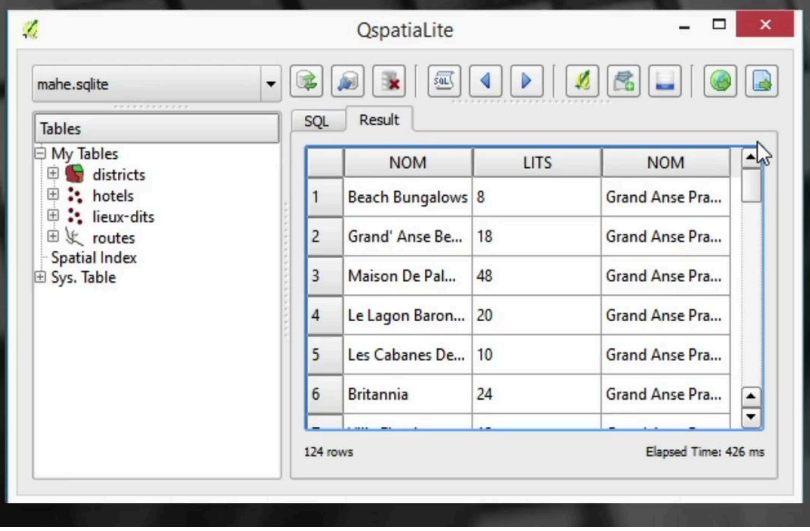
So the SELECT keyword, the NAME and BEDS fields, number of beds of the HOTEL table, the FROM keyword with the HOTEL table and then the conditional clause in which we express the idea that the district identifier which is associated with the hotel is equal to its equivalent in the DISTRICT table.

Notes

Summary

17m 03s

Joins based on the WHERE clause



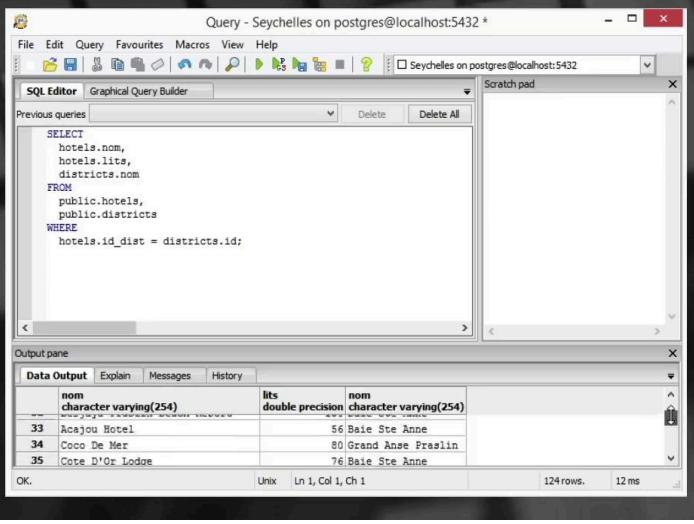
We add the district name to the desired result and then the name of the DISTRICT table. And we get the list of the 124 hotels of the Seychelles with their number of beds and the name of the district in which they are located.

Notes

Summary

17m 26s

Joins based on the WHERE clause



The same operation now in the pgAdmin interface of the postgres postGIS table where we graphically link the 2 tables by the identifier field, we select the fields which we want to see appearing in the result and we execute the query directly.

Notes

Summary

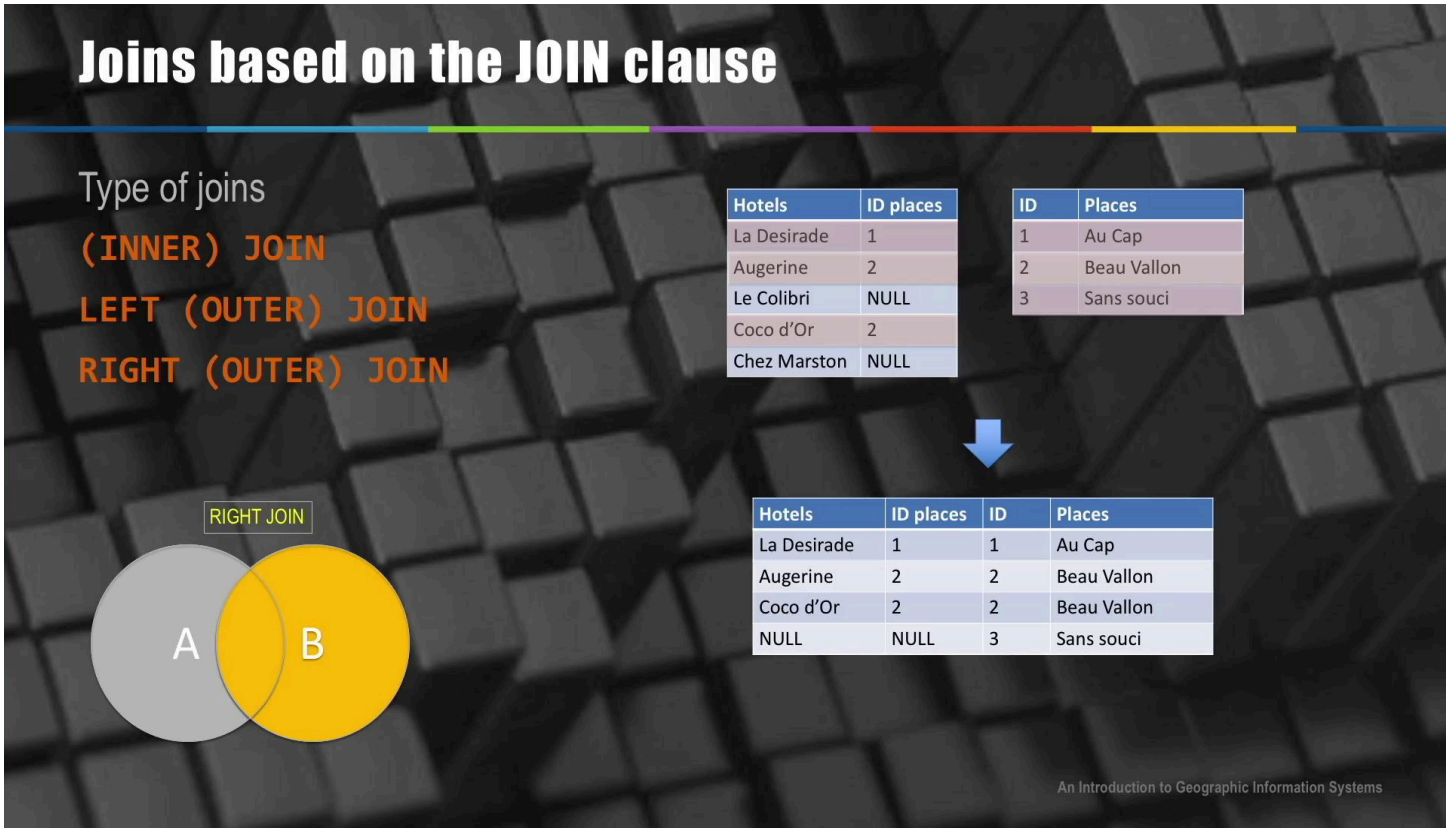
17m 40s

The basic syntax of a join query based on the JOIN clause is as follows: so first the SELECT keyword, the relevant attributes which we want to see appearing in the result, so an attribute of the table 1, an attribute of the table 2. The join that goes from table 1 to table 2 and then the ON keyword to specify the field on which this join is done in this case the attribute 3 of table 1 which is connected with the attribute 4 of table 2. If we compare this syntax with the one we have just seen in the case of the join based on the WHERE clause, we see that the difference is very small since the word JOIN has simply been introduced to separate the tables and the ON keyword instead of the WHERE clause to define the join criterion. The interest of using the syntax that uses the JOIN clause, is to separate well the join elements from the conditional elements in a complex SQL query, which makes the query more readable. There are several types of joins starting with the simple join or INNER JOIN the INNER keyword being unnecessary, INNER JOIN and JOIN are equivalent things.

Notes

Summary

18m 09s



Let's take the example of a series of hotels associated to a locality and of a second table in which there is a series of localities and we see that in our example the elements that will be selected, for which the "ID locality" and "ID" fields correspond, are 3, the 3 hotels La Desirade, Augerine and Coco d'Or which are found in the result table. Second type of join, the LEFT JOIN or LEFT OUTER JOIN, with the word OUTER which is again optional which consists in taking all the elements of table A, of set A, to which we join the corresponding elements of table B, so in our case all the hotels of the HOTEL table with, when it is possible, the joined information elements of the LOCALITY table. And we see that our result now contains 5 elements with null values for the joins in 2 cases and the joined parameters, so the locality. Third join type, the RIGHT JOIN which allows to select all the elements of the second table so all the localities to which we associate, when it is possible, the elements of the first table which would not be null at the join level. And we obtain as a result a table of 4 values including the 3 basic cases where the join exists and then the last case, the Sans souci" locality that has no join and for which the elements of the HOTEL table are null.

Notes

Summary

19m 48s

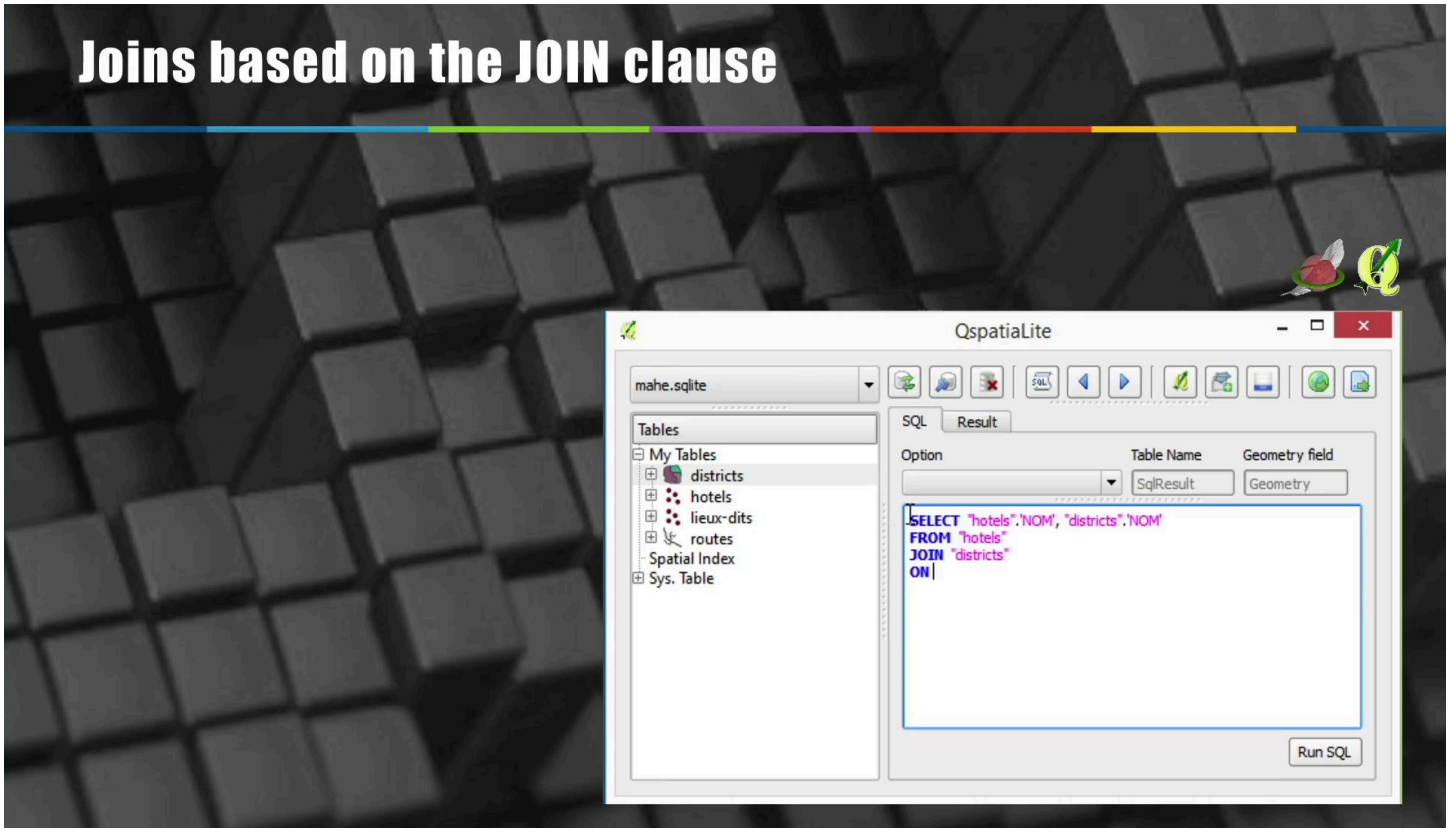
And finally the FULL JOIN which consists in taking all the possibles, so the series of 3 joins where we have objects linked on both sides plus the 3 cases where, either on side A or on side B, the joined elements are null. And we obtain a table that has 6 elements so 2 less than the table of 8 which we would have obtained if no joined element had been present. The syntax of these specific join queries still remains the same with just the LEFT, RIGHT or FULL clause which is added to the JOIN clause in the join definition.

Notes

Summary

21m 40s

Joins based on the JOIN clause



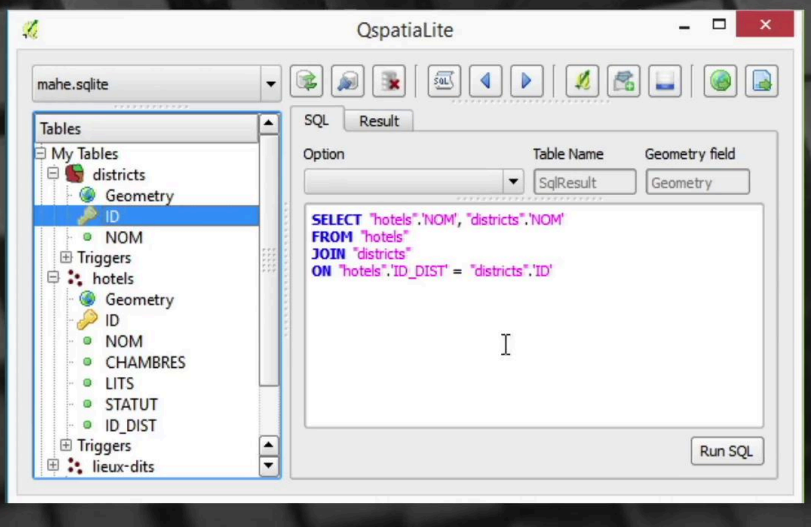
An example with the SpatiaLite database of the Seychelles where we define in the objects of the request the name of the hotels and then the name of the districts in which these hotels are situated that we will link by a join.

Notes

Summary

22m 31s

Joins based on the JOIN clause



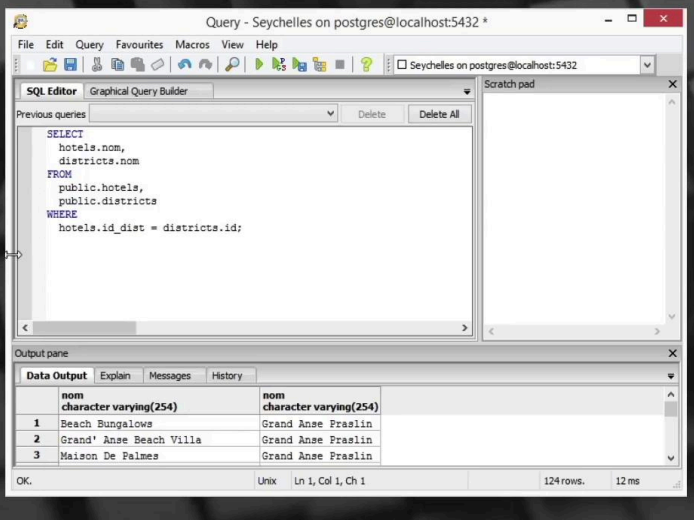
We define this join between the HOTEL table and the DISTRICT table on the basis of the district identifier field of the HOTEL table is equivalent to the district identifier in the DISTRICT table.

Notes

Summary

22m 53s

Joins based on the JOIN clause



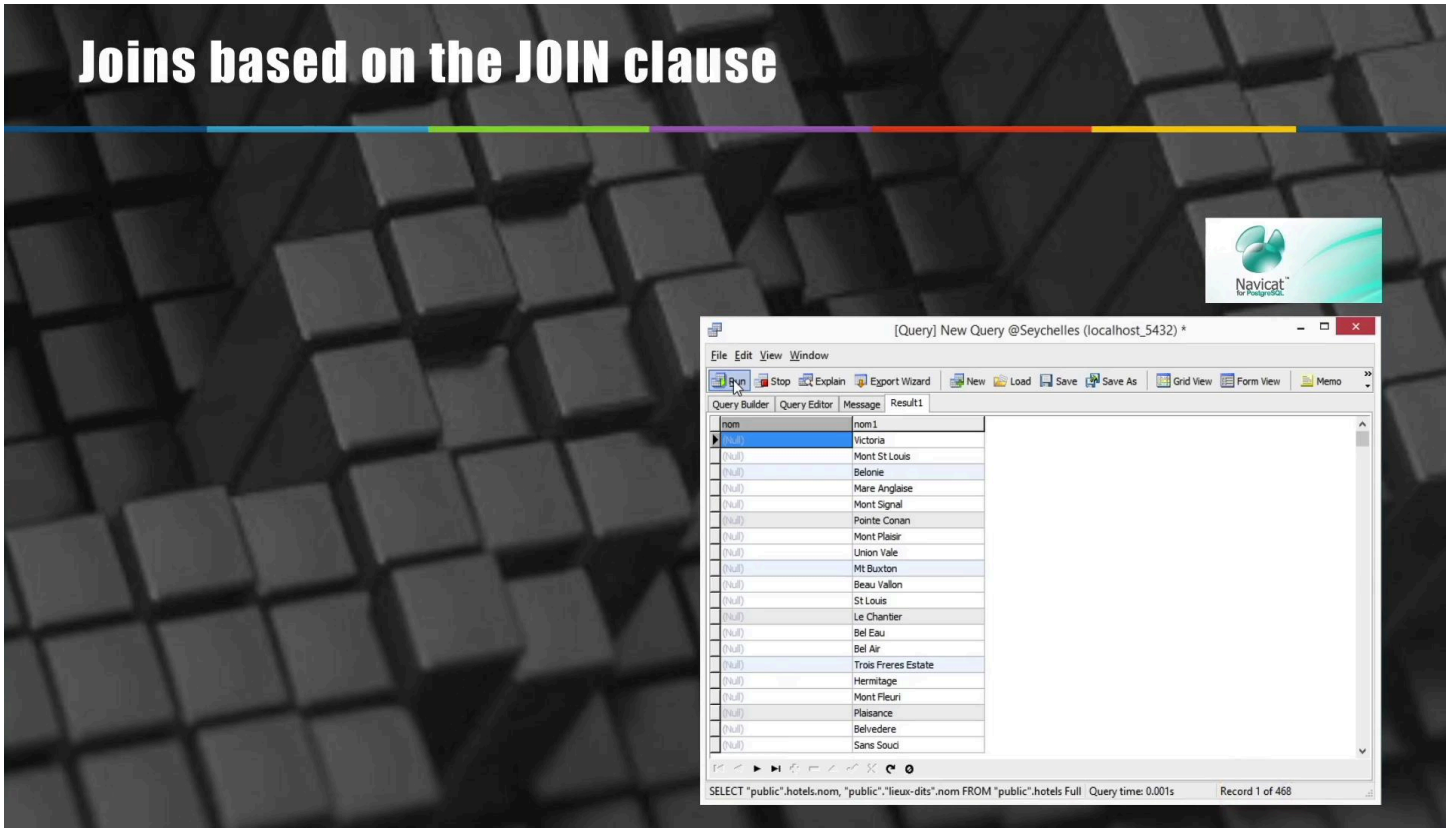
Same operation in postgres postGIS with pgAdmin where we graphically add the 2 tables in the graphical query constructor, we establish connection of these two tables, the fields we want to see appearing and in the join tab we define the join that interests us. However, we see that in the SQL edition part of the query, the query is written in a WHERE clause form and not a JOIN clause. I can rectify this by replacing the syntax and we see that the result is always the same. But if with this interface pgAdmin, I switch to graphic mode to return to editor mode, the query is transformed again into a WHERE query without keeping the JOIN keyword.

Notes

Summary

23m 05s

Joins based on the JOIN clause



Another example now with Navicat software which is a commercial software and which offers an interesting SQL query graphic constructor. We add the hotel table, this time the locality table, we establish a link between the name of the hotel and the name of the localities, we select these two elements, we see that we find 2 hotels whose name is the same as that of the localities. In the graphical interface. I can now replace the standard join by a LEFT JOIN which will give me all the 124 hotels with at the top of the list the 2 for which there is a locality in the join. Then we go to the RIGHT JOIN which will give me all the localities including the 2 joined hotels and we see that there are 346 of these localities and finally we can still do a FULL JOIN of these 2 tables to find that we have 466 lines in the answer.

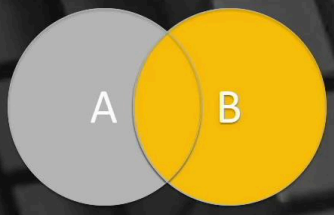
Notes

Summary

24m 01s

Joins based on the JOIN clause

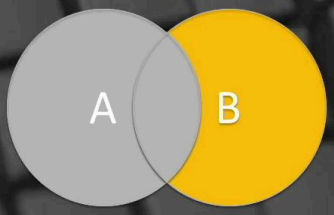
- RIGHT JOIN (sans l'intersection)



Hotels	ID places
La Desirade	1
Augerine	2
Le Colibri	NULL
Coco d'Or	2
Chez Marston	NULL

ID	Places
1	Au Cap
2	Beau Vallon
3	Sans souci

```
SELECT name_table1.name_attribute1,
name_table2.name_attribute2
FROM name_table1
RIGHT JOIN name_table2
ON name_table1.name_attribute3 =
name_table2.name_attribute4
```



Hotels	ID lieu-dit	ID	Lieu-dit
NULL	NULL	3	Sans souci

```
SELECT name_table1.name_attribute1,
name_table2.name_attribute2
FROM name_table1
RIGHT JOIN name_table2
ON name_table1.name_attribute3 =
name_table2.name_attribute4
WHERE name_table2.name_attribute3 IS NULL
```

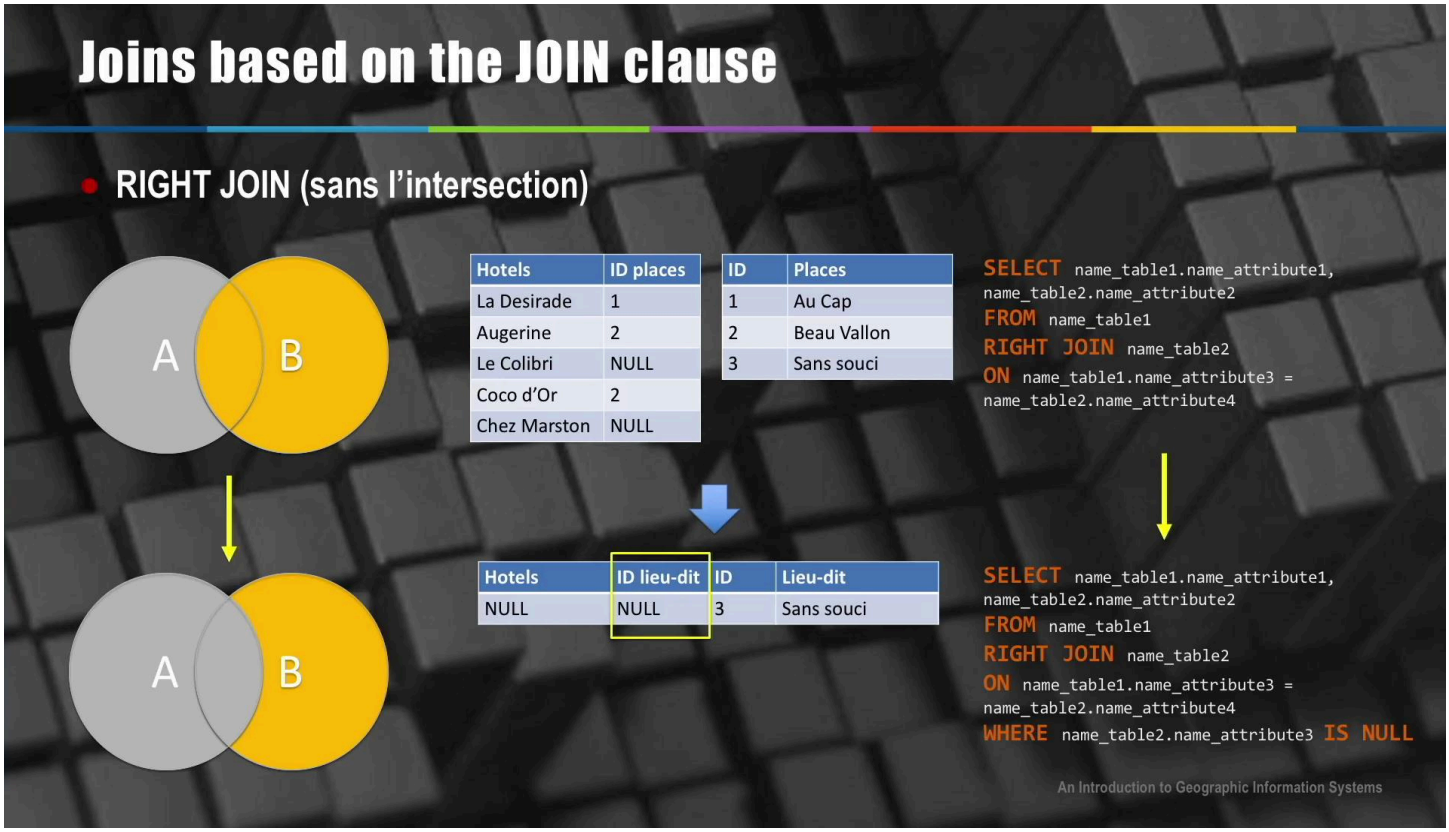
An Introduction to Geographic Information Systems

In these joins based on the JOIN clause we can look at a few particular cases which can be illustrated even if the use of these diagrams is not quite correct as we will see later when we will talk about merging requests. It nevertheless allows to illustrate the spirit of the subject so we have the LEFT JOIN request which elements that make the match we would like to remove to have only the elements of table A which have no correspondence in table B. If we take our example of 5 hotels and 3 localities we remember that the LEFT JOIN gave a table of 5 results and without the intersection it simply means removing the 3 cases where the join exists to keep only the 2 elements which are non-joining. From the point of view of the SQL syntax, it simply means that we will add a filter clause, a conditional filter to the join query in which we express the idea that the join attribute in this case the attribute 4 of table 2, so the ID field of the locality table is null. Same thing for the right join with in this case the 4 selected fields which, when the joined elements are removed, are reduced to a single result. And again in terms of SQL syntax the addition of a conditional clause based on the fact that the attribute of the join in the first table is null.

Notes

Summary

25m 12s



For the full request, we had 6 results from which we removed the 3 joined elements to obtain a final result that includes 3 objects. And from the SQL point of view, we add this time a conditional clause which expresses the fact that the 2 attributes of the join, that in the 2 attributes of the join, one or the other must be null.

Notes

Summary

27m 00s